ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Laboratorio di
Sicurezza Informatica

Offensive security I
Reconnaissance & Assessment

Marco Prandini

Dipartimento di Informatica – Scienza e Ingegneria

Offensive security?

- Porsi nel ruolo degli attaccanti
  - verificare l'esistenza di vulnerabilità
  - stimare con precisione l'impatto degli attacchi
  - testare l'efficacia delle contromisure
- Reconnaissance = primo anello della kill chain

  https://attack.mitre.org/tactics/TA0043/
- In questa parte del corso
  - enumerazione
  - scansione
  - brute forcing
  - vulnerability assessment

  limitatamente all'esposizione dei sistemi

Offensive security!

- Usare le stesse tecniche degli attaccanti è delicato
- MAI farlo su risorse non proprie senza permesso
- "permesso" è un termine da definire in modo ampio
  - semplici grattacapi da operazioni sospette
  - conseguenze legali
  - effetti imprevisti anche in buona fede
  - effetti sulle reti attraversate per raggiungere l'obiettivo lecito
- Scopo, efficacia ed efficienza dei test
  - velocità o precisione?
  - ricerca esaustiva di vulnerabilità o verifica della sensibilità dei sistemi di rilevazione?

Testing

- Fondamentale per
  - verificare se sono sfuggite vulnerabilità
  - verificare se il sistema è esposto a rischi nuovi rispetto al momento della progettazione
- Problema concettuale: copertura
  - Non su può dimostrare l'assenza di problemi
  - Solo tentare di sollecitare il sistema nel modo più completo possibile per trovare eventuali problemi esistenti
- Tre livelli di approfondimento
  - Vulnerability Assessment
  - Penetration Testing
  - Red Team Operations

VA

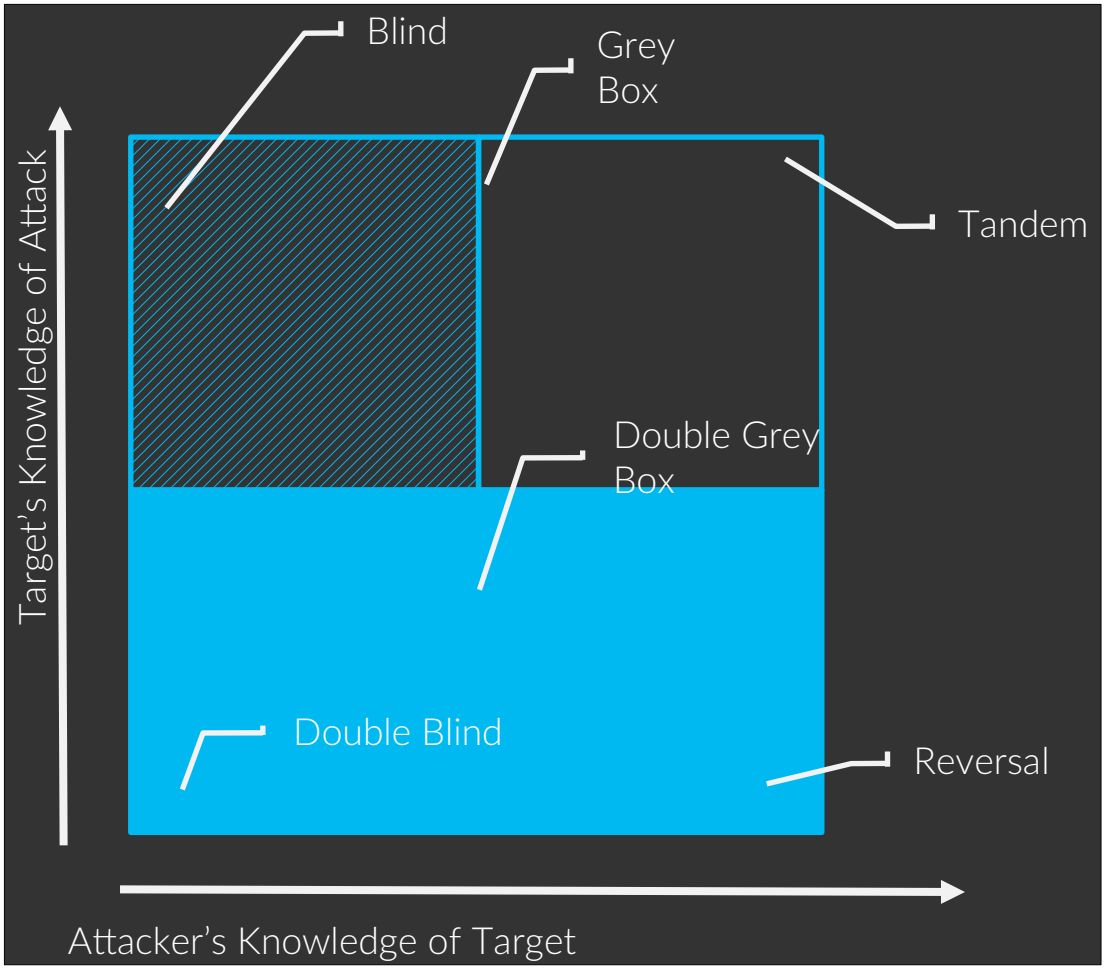
- La comunità pubblica le vulnerabilità scoperte, secondo un principio di *responsible disclosure*
- Esistono database human e machine-readable, es.
  - Common Vulnerabilities and Exposures http://cve.mitre.org/
  - National Vulnerability Database http://nvd.nist.gov/
  - Open Sourced Vulnerability Database http://osvdb.org/
  - SecurityFocus http://www.securityfocus.com/vulnerabilities
  - US-CERT http://www.kb.cert.org/vuls/
- Esistono software per cercarle sui sistemi
  - es. OpenVAS – dettagli in seguito
- Esistono database di exploit pronti per sfruttarle
  - https://www.cvedetails.com/
  - https://www.exploit-db.com/
  - https://packetstormsecurity.com/

VA → PT

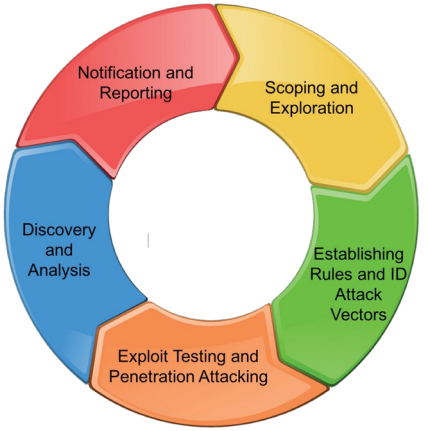
- VA trova solo vulnerabilità note
- Non procede oltre
  - Sfruttando una vulnerabilità si potrebbe accedere a una vista più interna e approfondita del sistema, svelandone altre
- Non considera la specificità del sistema
  - Anche falsi positivi, es. servizi che dichiarano una versione vulnerabile ma sono stati corretti
- PT: il tester (umano) avanza fin dove può, sfruttando le vulnerabilità per mezzo di exploit
  - Più realistico
  - Report più dettagliato
  - RISCHIOSO

PT - punti di partenza

- **Valutazione del target**
  - vengono stabilite le regole di ingaggio
  - mappatura, prioritizzazione, tracciamento dei confini
- **Postura e visibilità**
  - gli attacchi ciechi possono sembrare più realistici, ma fanno solo perdere tempo al tester esperto che è meglio spendere sui dettagli veramente nascosti
- **Protezione del bersaglio**
  - dove possibile, viene creata una replica per evitare di danneggiare il bersaglio, ma...
  - alcuni sistemi sono semplicemente troppo complessi
  - alcuni sistemi sono troppo critici per rischiare di perdere qualche dettaglio nella replica che potrebbe alterare il test



PT - metodologie



- Seguire una metodologia consente di
  - assicurarsi che il test sia coerente e ripetibile
  - eseguire una misurazione accurata della sicurezza (nessun pregiudizio o ipotesi o prove aneddotiche)
- Esistono alcune metodologie generalmente accettate:
  - Open Source Security Testing Methodology Manual (OSSTMM)
    - consente a qualsiasi tester di sicurezza di fornire idee per eseguire test di sicurezza più accurati, attuabili ed efficienti.
    - consente la libera diffusione delle informazioni e della proprietà intellettuale
- Open Web Application Security Project (OWASP)
  - specifico per applicazioni web
- Payment Card Industry Data Security Standard (PCI DSS)
  - settore finanziario; la sezione 11.3 riguarda il pentesting
- Technical Guide to Information Security Testing and Assessment (NIST800-115)
  - uno standard ufficiale del governo degli Stati Uniti
- Information Systems Security Assessment Framework (ISSAF)
  - completo ma non sviluppato attivamente

Preparazione

- Reconnaissance
  - raccolta di informazioni utili
  - estensione del perimetro di test
  - preparazione degli strumenti
- Enumeration
  - delimitazione del perimetro di test
  - verifica puntuale delle risorse e delle loro proprietà

OSINT

- Open Source INTelligence
  - L'uso di qualsiasi fonte pubblicamente disponibile per ricavare informazioni su di uno specifico obiettivo
  - Un campo di applicazione più ampio rispetto alla cybersecurity!
- OSINT su altri (vedremo più avanti)
  - componente della threat intelligence
  - componente dell'incident response
- OSINT su se stessi
  - cosa possono scoprire gli avversari?
  - come possono essere usate queste informazioni?
- È legale?
  - sostanzialmente sì
  - attenzione alle aree grigie

    https://mediasonar.com/2020/03/11/10-tips-for-doing-osint-legally/

OSINT – strumenti e fonti online

- https://osintframework.com/
- Ad esempio, per misurare l'esposizione dell'infrastruttura
  - collocazione fisica
    - geolocation
    - rilevazione di indirizzi da documenti e pagine web
  - collocazione in rete
    - domini DNS associati all'obiettivo
    - range di indirizzi IP
    - provider di connettività e autonomous systems
    - certificati X.509
  - accesso ai servizi
    - porte raggiungibili
    - fingerprinting dei sistemi → anche notoriamente vulnerabili
    - username validi → anche con relative password

Internet in una slide, ai fini dell'enumeration

- Ogni host bersaglio ha un indirizzo IP
  - I blocchi di IP sono assegnati dallo IANA ai RIR

    https://www.iana.org/numbers
  - I RIR li sub-allocano ai LIR, tenendo traccia di chi è l'effettivo responsabile di ogni blocco
- Oltre agli indirizzi, sono molto utili i nomi DNS
  - Ancora lo IANA coordina la concessione dei Top Level Domains

    https://www.iana.org/domains/root/db
  - Innumerevoli *registrar* si occupano della registrazione dei domini di secondo livello
- Ogni host raggiungibile via IP può esporre punti di accesso
  - Funzioni di base del protocollo IP gestite direttamente dal sistema operativo (es. rispondere al *ping*)
  - Processi in ascolto (listen) su porte TCP o UDP

Raccolta di informazioni – DNS

- I record DNS possono svelare
  - gli IP registrati dall'obiettivo
  - l'esistenza e la collocazione di specifici server applicativi
  - l'esistenza di sottoreti non direttamente raggiungibili
  - alias per sistemi collocati al di fuori del perimetro dell'obiettivo
    - risorse in cloud
    - sistemi legati da relazioni di fiducia es. domini di una foresta Active Directory
- Questo consente un notevole risparmio di tempo rispetto alla forza bruta
- Aggravanti
  - plateali: abilitazione di domain transfer
  - sottili: permanenza di record rimossi nelle cache
- Strumenti
  - lookup di base: `host`, `dig`, `nslookup`
  - strumenti di ricerca che includono guessing e forza bruta: `dnsenum`, `dnsmap`, `dnsrecon`, `fierce`, ...

Raccolta di informazioni – IP blocks

- La conoscenza dei dettagli organizzativi o di pochi indirizzi IP validi può permettere di espandere la conoscenza
  - agli interi blocchi allocati all'obiettivo
  - ad altri blocchi non evidentemente collegati
- Esempio
  - da www.unibo.it riuscite a risalire a tutte le reti degli enti di ricerca e delle università italiane?
  - hint: strumenti di ricerca RIPE

Enumerazione – host

- Una volta individuati i blocchi di indirizzi da analizzare si procede con l'individuare gli host effettivamente attivi (live host)
- Banale `ping`
  - 1 indirizzo per volta
  - bloccato da router e firewall?
  - ignorato da host?
    - scansione mirata ai servizi (in seguito)
- Scansioni massive
  - `masscan`
- Su rete locale più strumenti
  - sniffing passivo (`wireshark`, `tshark`, `tcpdump`, ...)
  - `arping`

Enumerazione – servizi

- Determinati gli host raggiungibili, si cercano le porte aperte
  - le due fasi possono collassare in una, se si sospetta che gli host interessanti ignorino i ping → test di vitalità fatto direttamente sondando le porte
- Il tool più diffuso: `nmap`
  - scansione contemporanea di range di indirizzi e porte
    - set predefinito di porte "più popolari"

      https://nmap.org/book/port-scanning.html
  - diverse tipologie di scansione
  - fingerprinting del sistema operativo e delle versioni dei servizi
- Alcuni vantaggi di `unicornscan` su `nmap`
  - fingerprinting più affidabile
  - relativamente più veloce
  - può salvare le risposte per analisi con altri strumenti

Evasione

- Scopi della fase di OSINT ed enumerazione
  - testare l'efficacia delle difese (IDS)
  - preparare il terreno per condurre un test della massima accuratezza evitando di incorrere in impedimenti (IPS)

  → **Reconnaissance in modalità anonima**
  - ToR o simili
  - creazione di account usa e getta sui siti
  - utilizzo di VM diverse e periodicamente sostituite

  → **Enumerazione in modalità "stealth"**
  - temporizzazione configurabile
  - randomizzazione di indirizzi e porte
  - non utilizzare lo stack TCP/IP dell'host di origine (unicornscan) per evitare reverse fingerprinting

  → **Enumerazione adattativa rispetto a FW, IDS e IPS**

  https://nmap.org/book/firewalls.html

Tentativi di accesso ai servizi

- Analisi dei protocolli applicativi più comuni
  - SMB, SMTP, SNMP
  - può portare ad accesso a dati o a raccolta di ulteriori informazioni per le fasi successive
- Brute forcing applicativo (fuzzing)
  - invio di payload randomizzati per tentare di sollecitare risposte impreviste (Es. `bed`, `doona`, vari tool per SIP, …)
- Framework per lo sviluppo e l'esecuzione di exploit

La postura interna

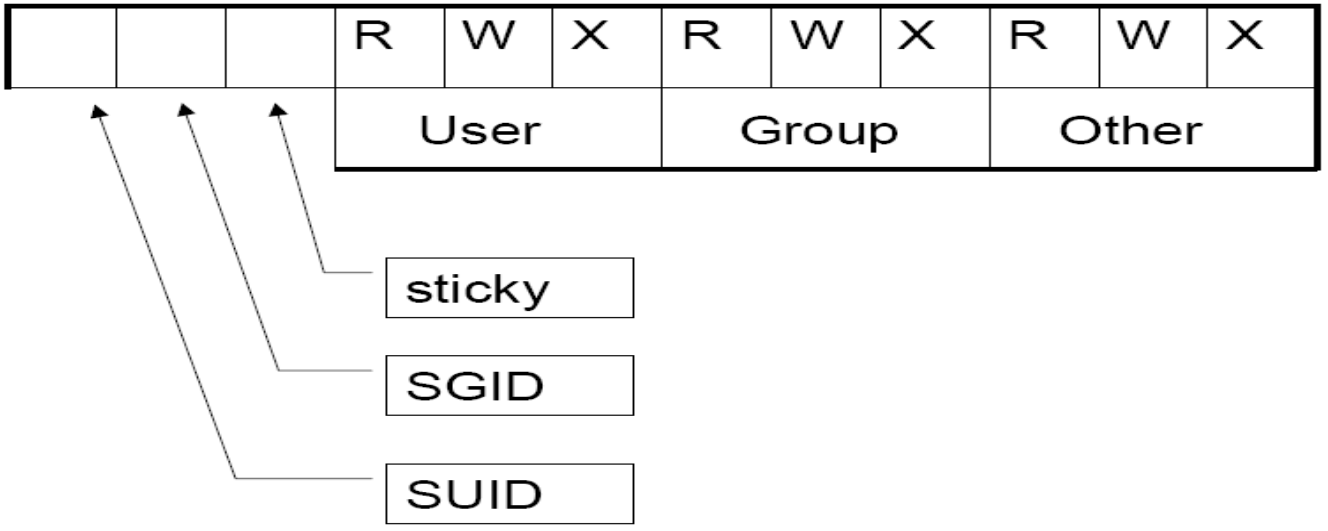
- Reconnaissance ed enumeration sono svolte da una postura esterna al perimetro di difesa dell'obiettivo
  - più fedele alla postura dell'attaccante
  - ma se non catturano un metodo di intrusione?
- Possibilità: auto-attaccarsi dalla posizione più vantaggiosa, all'interno
  - NIDS e FW tipicamente scavalcati
  - sistemi raggiunti senza ostacoli
    - miglior test per HIDS
    - maggior efficienza
- Verifiche tipiche (oltre a quanto visto)
  - controllo dell'accesso (utenti, permessi)
  - esposizione servizi in rete
  - iniezione di software e occultamento di processi

Reti

- Wireless: un modo eccellente di guadagnare una postura interna, se ci si trova fisicamente a portata di segnale
  - accesso alle comunicazioni
  - possibilità di attacco dei sistemi raggiungibili
- WiFi: http://www.aircrack-ng.org/
  - cattura di pacchetti per analisi successiva
  - attacchi di tipo replay e deauthentication
  - creazione di finti access point
  - cracking della chiave per WEP, WPA1-PSK, WPA2-PSK
- Non solo WiFi
  - bluetooth, NFC, SDR in generale
- Post-accesso wireless o fisico
  - toolkit per l'esecuzione di MITM

Ripasso autorizzazioni su Unix Filesystem

- Ogni file (regolare, directory, link, socket, block/char special) è descritto da un i-node
- Un set di informazioni di autorizzazione, tra le altre cose, è memorizzato nell'i-node
  - (esattamente un) utente proprietario del file
  - (esattamente un) gruppo proprietario del file
  - Un set di 12 bit che rappresentano permessi standard e speciali



Significato dei bit di autorizzazione

- Leggermente diverso tra file e directory, ma in gran parte deducibile ricordando che
  - Una directory è semplicemente un file
  - Il contenuto di tale file è un database di coppie (nome, i-node)

R = read (lettura del contenuto)

Lettura di un file

Elenco dei file nella directory

W = write (modifica del contenuto)

Scrittura dentro un file

Aggiunta/cancellazione/rinomina di file in una directory

X = execute

Esegui il file come programma

Esegui il lookup dell'i-node nella

NOTA che il permesso 'W' in una directory consente a un utente di cancellare file sul contenuto dei quali non ha alcun diritto

NOTA: l'accesso a un file richiede il lookup di tutti gli i-node corrispondenti ai nomi delle directory nel path → serve il permesso 'X' per ognuna, mentre 'R' non è necessario

SUID e SGID

- Supponiamo che un utente U, che in in dato momento ha come gruppo attivo G, lanci un programma
- Il processo viene avviato con una quadrupla di identità:
  - real user id (ruid) = U
  - real group id (rgid) = G
  - effective user id (euid) identità assunta dal processo per operare come soggetto diverso da U
  - effective group id (egid) identità di gruppo assunta dal processo per operare come soggetto diverso da G
- Normalmente euid=ruid e egid=rgid
- Alcuni permessi speciali attribuiti a file eseguibili possono fare in modo che euid e/o egid siano diversi dai corrispondenti ruid / rgid
  - si definiscono programmi Set-User-ID o Set-Group-ID

Bit speciali / per i file

I tre bit più significativi della dozzina (11, 10, 9) configurano comportamenti speciali legati all'utente proprietario, al gruppo proprietario, e ad altri rispettivamente

- BIT 11 – SUID (Set User ID)
  - Se settato a 1 su di un programma (file eseguibile) fa sì che al lancio il sistema operativo generi un processo che esegue con l'identità dell'utente proprietario del file, invece che quella dell'utente che lo lancia
- BIT 10 – SGID (Set Group ID)
  - Come SUID, ma agisce sull'identità di gruppo del processo, prendendo quella del gruppo proprietario del file
- BIT 9 – STICKY
  - OBSOLETO, suggerisce al S.O. di tenere in cache una copia del programma

Bit speciali / per le directory

- Bit 11 per le directory non viene usato
- Bit 10 – SGID
  - Precondizioni
    - un utente appartiene (anche) al gruppo proprietario della directory
    - il bit SGID è impostato sulla directory
  - Effetto:
    - l'utente assume come gruppo attivo il gruppo proprietario della directory
    - I file creati nella directory hanno quello come gruppo proprietario
  - Vantaggi (mantenendo umask 0006)
    - nelle aree collaborative il file sono automaticamente resi leggibili e scrivibili da tutti i membri del gruppo
    - nelle aree personali i file sono comunque privati perché proprietà del gruppo principale dell'utente, che contiene solo l'utente medesimo
- Bit 9 – Temp
  - Le "directory temporanee" cioè quelle world-writable predisposte perché le applicazioni dispongano di luoghi noti dove scrivere, hanno un problema: chiunque può cancellare ogni file
  - Questo bit settato a 1 impone che nella directory i file siano cancellabili solo dai rispettivi proprietari

Controllo dell'accesso

- Rischi associati ai permessi
  - violazione diretta della sicurezza dei dati (file leggibili o modificabili da troppi soggetti)
    - permessi standard
    - POSIX ACL
  - privilege escalation
    - programmi con bit SUID impostato: generano un processo che assume l'identità dell'utente proprietario del file, invece che dell'utente che ha lanciato il comando
    - bit SGID impostato: idem, con l'identità di gruppo
    - capabilities
- Diverse cause di privilege escalation
  - programmi che non dovrebbero avere privilegi speciali
  - programmi che hanno necessità lecita di privilegi speciali ma contengono vulnerabilità
    - utilizzabili per scopi diversi da quelli di progetto
  - file che vengono utilizzati in modo insicuro da processi privilegiati
    - Corse critiche, TOCTOU

Permessi e ACL

- Ricerche tipiche di file che possono causare problemi
- Bit SUID/SGID

  ```
  find / -type f -perm /6000
  ```

- File scrivibili da tutti

  ```
  find / -perm /2
  ```

- File che non sono di proprietà di alcun utente valido

  ```
  find / -nouser
  ```

- Per le ACL è più complicato, si può partire da

  ```
  getfacl -R /
  ```

Capabilities

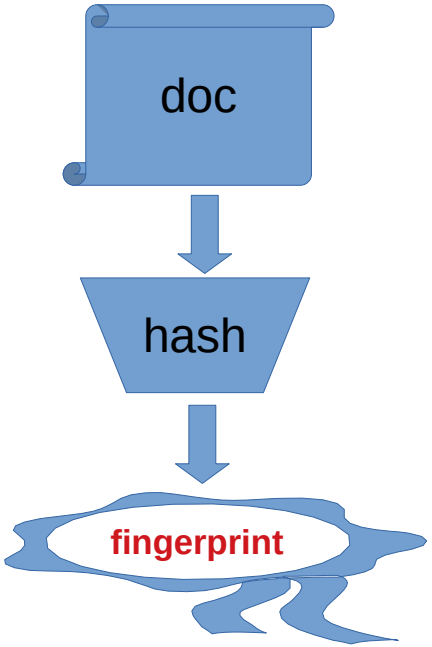
- Le capabilities sono ciò che distingue root dagli utenti standard
  - root le ha tutte (~40)
  - possono essere assegnate singolarmente a un processo per mezzo di attributi estesi associati al programma
- Vediamole per capire quali sono le più pericolose

  https://man7.org/linux/man-pages/man7/capabilities.7.html
- Ricerca di file con capabilities settate:

  ```
  getcap -r /
  ```

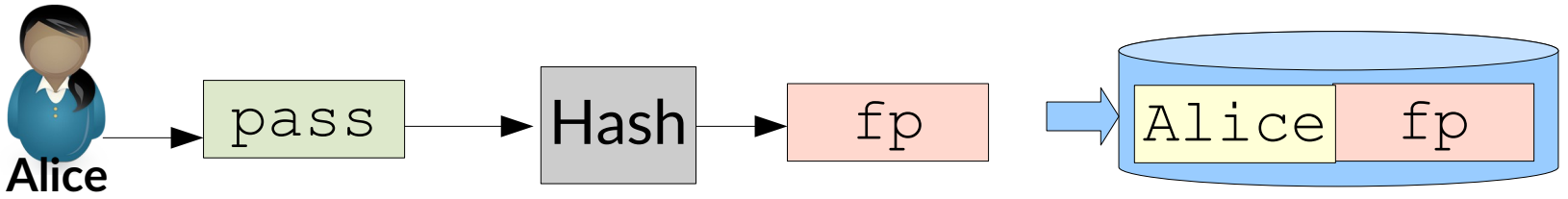
Utenti

- Le credenziali degli utenti sono memorizzate in file del sistema
  - protetti dai permessi
  - contenenti non le password in chiaro, ma le loro *impronte* generate da un algoritmo di *hash*
- Approfondiremo il tema quando parleremo di crittografia, per ora sintetizziamo e accettiamo "a scatola chiusa" che una funzione hash
  - Produce un'impronta di dimensione fissa a partire da un input arbitrario (quindi non è direttamente invertibile)
  - È costruita in modo che dedurre l'input originale dall'impronta sia pressochè impossibile
  - È costruita in mdo che produrre due documenti che abbiano la stessa impronta sia pressoché impossibile

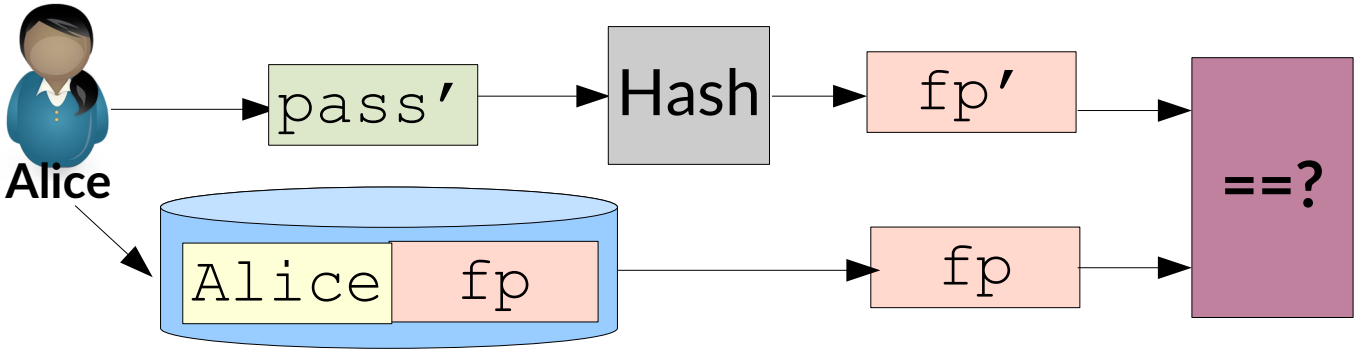


Password - semplificato

- Scelta della password



- Verifica della password



- Se le ipotesi sulla funzione hash sono corrette, non c'è modo più efficiente per dedurre una password che tentare di indovinarla

Attacco alle credenziali utente

- Il più classico degli assessment: robustezza delle password
- Nella posizione più vantaggiosa (root) si potrebbero impersonare tutti gli utenti
  - perché rubare password?
  - utilizzo frequente su altri sistemi!
- Password cracking a forza bruta
  - interattivo → lento, tendente all'impossibile
  - avendo gli hash → ricerca con rainbow tables
    http://project-rainbowcrack.com/
    - compromesso spazio-tempo da valutare
- Password cracking con dizionario
  - John the ripper https://www.openwall.com/john/
    - wordlist enormi disponibili online
  - costruzione di wordlist su misura per caratteristiche note dell'obiettivo
    https://github.com/Mebus/cupp
    https://github.com/digininja/CeWL

Normalmente contenuti in file non leggibili dall'utente standard, ma potrebbero essere esfiltrati se è presente una vulnerabilità che consente una privilege escalation

File e socket accessibili

- `lsof` lists open files
  - TCP and UDP sono solo namespace differenti

```
# lsof -i -n | egrep 'COMMAND|LISTEN|UDP'
COMMAND    PID USER   FD   TYPE DEVICE SIZE NODE NAME
sshd      2317 root    3u  IPv6   6579       TCP *:ssh (LISTEN)
xinetd    2328 root    5u  IPv4   6698       TCP *:auth (LISTEN)
sendmail  2360 root    3u  IPv4   6729       TCP 127.0.0.1:smtp (LISTEN)
```

  - provate `lsof | grep '(deleted)'`

Punti di acccesso esposti via rete

- `netstat` mostra lo stato delle socket Unix e di rete
  - di default, già connesse a un altro endpoint
    - opzione `-l` : listening
    - opzione `-a` : entrambe le categorie
  - altre opzioni utili
    - `-p` processo in ascolto
    - `-n` output numerico
    - `-t` tcp socket
    - `-u` udp socket
- `ss` è il rimpiazzo più recente, però non è SELinux-aware

netstat sample output

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:2049            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:993             0.0.0.0:*               LISTEN      3457/inetd
tcp        0      0 0.0.0.0:901             0.0.0.0:*               LISTEN      3457/inetd
tcp        0      0 0.0.0.0:904             0.0.0.0:*               LISTEN      11325/rpc.mountd
tcp        0      0 0.0.0.0:3689            0.0.0.0:*               LISTEN      11438/mt-daapd
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      20600/mysqld
tcp        0      0 0.0.0.0:3690            0.0.0.0:*               LISTEN      11441/mt-daapd
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN      3717/smbd
tcp        0      0 0.0.0.0:110             0.0.0.0:*               LISTEN      3457/inetd
tcp        0      0 0.0.0.0:143             0.0.0.0:*               LISTEN      3457/inetd
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN      2953/portmap
tcp        0      0 0.0.0.0:6001            0.0.0.0:*               LISTEN      14660/Xrealvnc
tcp        0      0 0.0.0.0:113             0.0.0.0:*               LISTEN      3457/inetd
tcp        0      0 137.204.58.80:993       137.204.58.138:51929    ESTABLISHED 8190/imapd
```

Processi a orologeria

- Sono un modo per garantire persistenza
  - presenti tra i processi solo quando necessario
  - riavviati dopo terminazione o reboot
- Eseguiti periodicamente
  - crontab di ogni utente
  - crontab di sistema
- Accodati per l'esecuzione ritardata
  - spool del demone atd

Iniezione di software

- Non banale ma estremamente impattante
- I sistemi di package management prendono, di default, sempre l'ultima versione di ogni pacchetto
- Aggiungere repository è un rischio
  - spesso lo si fa in modo legittimo per installare un'applicazione magari semisconosciuta ma innocua
  - un pacchetto messo nel repository "minore" potrebbe sostituirne uno cruciale con lo stesso nome
    - repo meno sorvegliati
    - repo senza firma digitale → MITM
- Verificare se esiste l'opportunità di iniettare pacchetti
  - file di configurazione delle sorgenti
  - keyring per la verifica delle firme
  - utilizzo dei tool di package management della distribuzione

Strumenti di ricerca locali

- Una miriade di altre possibili vulnerabilità o semplici informazioni utili per test più efficaci
- Servono strumenti di scansione approfondita, es.
  https://github.com/rebootuser/LinEnum
  - Informazioni sul sistema
  - Informazioni sugli utenti
  - Esecuzione automatica di programmi
  - Servizi installati e in esecuzione
- Esempi di impostazioni insicure riportate
  - default umask
  - permessi e capabilities
  - dati sensibili nella history, env vars, …
  - credenziali di default
- Non necessariamente coprono tutto!

Strumenti di ricerca completi

- Esistono scanner di vulnerabilità completi
  - configurabili per eseguire la scansione di una combinazione arbitraria di host e porte
  - possono testare l'esistenza di vulnerabilità a livello di rete, sistema operativo e applicazione tramite plug-in caricabili
  - possono collegare ogni vulnerabilità alla documentazione pertinente (ad esempio CVE)
- Il più noto è Nessus, un prodotto commerciale di Tenable, attualmente alla versione 8
  - www.nessus.org
- OpenVAS è l'open-source equivalente, essendo un fork di nessus 2.2 avviato nell'agosto 2008 e attivamente sviluppato da allora
  - www.openvas.org

OpenVAS – caratteristiche principali

- architettura
  - scan engine (svolge i test)
  - manager (coordina i task di scansione)
  - interfaccia (pianifica i task per il manager, mostra i risultati)
  - amministrazione (gestisce utenti e database)
- si appoggia su di un vulnerability database
  - Network Vulnerability Tests
    - descrizione della vulnerabilità
    - piattaforme colpite
    - processo di verifica
  - aggiornato ogni giorno!